

Removal-Cost Method: An Efficient Voltage Selection Algorithm for Multi-Core Platforms under PVT

Sohaib Majzoub, Resve Saleh, Steven J.E. Wilton, and Rabab Ward

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada

ABSTRACT

In this paper, we present a novel solution to the Voltage Selection Problem for large multi-core architectures. Compared to previous algorithms, ours provides similar results, but is more than 10x faster. This run-time improvement is important, especially for large multi-core platforms with hundreds of cores. We evaluate our algorithm in the context of a process, voltage, and temperature (PVT) variation-aware energy optimization framework.

I. INTRODUCTION

Multiprocessor System-on-Chip (MPSoC) architectures have emerged as a computing platform that provides high-performance and flexibility. A MPSoC consists of hundreds or thousands of simple processors on a single chip connected using a structured network. These architectures are especially suitable when multiple applications are running simultaneously, each with multiple tasks [1].

One of the challenges of such architectures is the power consumption. Thus, MPSoC's with large number of active processors results in significant switching and leakage power. Consequently, power consumption can make such platforms unsuitable for mobile devices with limited battery lifetimes. In addition, the thermal effects, due to power consumption, can lead to packaging problems [1][2][3][4].

An effective way of reducing power is to have multiple supply voltages in the design. Therefore, cores, running non-timing-critical tasks, are assigned relatively low supply voltage, hence, they consume less power but they run slower. On the other hand, cores, running timing-critical tasks, are assigned relatively high supply voltages. Thus, they consume more power but they run faster [5][6][7][8].

Theoretically, each processor could be assigned its own voltage. However, a large number of voltages leads to significant overhead in the power distribution network. Thus, it is important to minimize the number of voltage levels in the design. For instance, consider using m different voltage levels during early optimization, however only d levels can actually be implemented, where $d < m$. Selecting d out of m is called the Voltage

Selection Problem (VSP) and is known to be NP-Hard [2][6][7][8].

Algorithms solving VSP have been developed to provide acceptable, but not provably optimal solutions. However, these previous heuristics can become very slow for MPSoC's with tens to hundreds of cores. In this paper, we present a new heuristic to solve VSP that is more than 10x faster than previous approaches, but provides similar results as the previous algorithms. This algorithm is called the Removal-Cost Method (RCM) and uses a single pass of the task graph to assign voltage levels. The use of such a fast algorithm will become critical as larger MPSoC's are built.

Complicating the situation is the presence of Process, Temperature, and Voltage (PVT) variations, which are common in nanoscale technologies [5]. These variations can be either within-die or die-to-die, and can be random or systematic. Within-die variations can cause an asymmetric frequency distribution among cores [9]. This asymmetry must be taken into account during the energy optimization. Thus, we present a PVT-aware energy optimization flow that includes the voltage selection problem to show that the RCM technique provides acceptable power reduction results in much less time than previous techniques.

II. RELATED WORK

Several previous papers have described algorithms to solve VSP [2][5][6][7][8]. Yan et al in [3] proposed a scheduling algorithm for a heterogeneous multiprocessor system that simultaneously uses dynamic voltage scaling and adaptive body biasing. In [4], Ruggiero et al optimized the selection of the number of processing cores and their corresponding voltage and frequency using a functional, cycle-accurate simulation on a virtual SystemC-based platform. Leung and Tsui in [6] proposed a voltage island partitioning for tile-based network-on-chip platforms. They proposed 1-STEPLA algorithm as a greedy approach for voltage selection and voltage island formation. Ghosh and Sen in [7] formulated the VSP problem as a Quadratic Program and then reduced it to an Integer Linear Program. They used a greedy randomized heuristic to solve the voltage assignment problem, and obtained a near-optimum solution. Sengupta and Saleh, in [8], used dynamic

programming and were also able to get near optimal solution but at the expense of high computational cost.

These previous works are primarily focused on multicore platforms with relatively few cores. Thus, these algorithms are slow for type of architectures considered in this paper, which have hundreds or thousands of cores. In addition, these previous studies did not consider process, voltage and temperature variation.

III. REMOVAL COST METHOD

In this section, we define the voltage assignment problem and present our proposed Removal Cost Method.

A. Problem Formulation

When an algorithm is mapped to an MPSoC, critical path analysis can be performed to determine the *slack* of each task. The slack is the amount the task can be slowed down before increasing the overall execution time of the algorithm. Once these tasks are mapped to physical cores, the voltage levels of each core can be adjusted to be as small as possible such that all tasks assigned to that core meet their timing constraints.

If the voltage of each core is adjusted independently, there will be m distinct voltage levels such that $m \leq N$ where N is the number of cores. If m is large, it is not practical to implement so many distinct voltages. In that case, a subset of the m voltages must be selected, such that each core can be powered by one of the selected voltages. Since there are many possible subsets, the Voltage Selection Problem is to select the subset that results in the minimum overall energy with a desired small number of voltage levels across the entire MPSoC. More precisely, the VSP is:

- Input:** a) A set of possible voltage levels: $X=\{V_1, V_2, \dots, V_m\}$
b) An initial assignment of voltages to cores: $v_0(i) \in X$ with $1 \leq i \leq N$ where N is the number of physical cores.
c) A target number of voltage levels d .

Find: A final assignment of voltages to cores, $v_f(i) \in X$, $1 \leq i \leq N$

- Such that:** a) The delay constraint is no worse than with the initial voltage assignment
b) The overall power is minimum.
c) The number of unique voltage levels is d

The delay constraint can be ensured by enforcing $v_f(i) \geq v_0(i)$, $1 \leq i \leq N$.

B. Initialization

The algorithm described in the next section requires a matrix E with N rows and m columns. Element $E_{i,j}$ is the amount of energy required by core i if implemented using voltage level V_j . In our implementation, we compute this by summing the energy dissipated by the

cores and the energy dissipated in the core C_i communication links as follows:

$$E_{i,j} = e_{bit} + D \times P_{core} \quad (1)$$

where e_{bit} is the energy for all communications computed as:

$$e_{bit} = n_{hops} \times e_{S_{bit}} + (n_{hops} - 1) \times e_{L_{bit}} \quad (2)$$

where n_{hops} is the number of utilized switches during transmission (or ‘‘hops’’), while e_S and e_L are the energy consumed by the switch and the link, respectively [5][6]. The power dissipated in the core is computed using the standard dynamic and static power equations as follows:

$$P_{core} = C_{eff} V_{dd}^2 f + I_o e^{-\frac{qV_t}{nkT}} V_{dd} \quad (3)$$

where C_{eff} is the effective switching capacitance, V_{dd} is the selected operating supply voltage, f is the frequency at which the core is running, T is the temperature, and I_o , and n are technology-dependent constants. The delay of the core can be written as:

$$D = \frac{C_o V_{dd} / 2}{K (V_{dd} - V_t)^\alpha} \quad (4)$$

where C_o is the switching capacitance of the critical path, K and α are technology-dependent constants.

C. RCM Algorithm

Figure 1 shows the RCM algorithm. The algorithm maintains an ordered list of selected voltage levels X and a voltage assignment for each core Y . Initially, $Y = \{v_0(i), 1 \leq i \leq N\}$, and X is the set of unique voltages in Y . Each iteration, one element is greedily removed from X , and Y is updated appropriately. The algorithm ends when there are d elements in X .

Each iteration, the voltage to be removed from X is selected using a vector R . R contains the removal cost for each voltage. Each element of R corresponds to a voltage level in X and is equal to the increase in energy that would occur if that voltage level was removed, and all cores currently assigned that voltage level were instead assigned the next highest voltage level. More precisely,

$$R_j = \sum_{\substack{\text{all } n \text{ cores such that} \\ v_n = V_j; V_j \in X}} (E_{n,j+1} - E_{n,j}) \quad (5)$$

As an example of our method, consider the cost matrix shown in Figure 2(a). Assume that we require only 2 voltage levels, i.e. $d=2$. Initially, $X = \{V_1, V_2, V_3, V_4\}$ and $Y = \{(v_0(1)=V_1), (v_0(2)=V_1), (v_0(3)=V_1), (v_0(4)=V_2), (v_0(5)=V_1), (v_0(6)=V_2), (v_0(7)=V_3), (v_0(8)=V_4)\}$. The initial cost by summing the shaded regions is 37.1. The removal cost of V_1 is $R_1 = (8.3-8.1)+(4.7-4.3)+(7-6.3)+(3.8-3.5) = 1.6$. The removal cost of V_2 is $R_2=1.1$. The removal cost of V_3 is $R_3=0.6$. Thus the vector R is $R=\{1.6, 1.1, 0.6, \infty\}$. Based on that, V_3 should be removed because it has the minimum removal cost $R_3=0.6$. Thus $X=\{V_1, V_2, V_4\}$, $Y=\{(v(1)=V_1),$

$(v(2)=V_1), (v(3)=V_1), (v(4)=V_2), (v(5)=V_1), (v(6)=V_2), (v(7)=V_4), (v(8)=V_4))$ and $R=\{1.6, 1.1, \infty\}$. After removing V_3 , only R_2 needs to be updated since $R_4=\infty$. The same process will repeat again for iterations 2 and 3 as shown in Figure 2 (b). At the end, V_2 and V_4 will be selected. The final Y vector would be $Y=\{(v(1)=V_2), (v(2)=V_2), (v(3)=V_2), (v(4)=V_2), (v(5)=V_2), (v(6)=V_2), (v(7)=V_4), (v(8)=V_4))$ and $X=\{V_2, V_4\}$.

L#	Removal Cost Method	Cost
	Inputs: $X = \{V_1, V_2, \dots, V_m\}$, m voltage levels in X , $Y = \{v(i), 1 \leq i \leq N\}$, cores' initial voltage assignment, Outputs: $X = \{V_1, V_2, \dots, V_d\}$, d voltage levels in X $Y = \{v(i), 1 \leq i \leq N\}$, cores' final voltage assignment	
	begin	
1	$R_j = 0$; for $1 \leq j \leq m$; /* Initialize vector R to zero */	$O(m)$
2	for $i = 1$ to N do	
3	$j =$ index of element V_j such that $v(i) = V_j$	$O(N)$
4	$R_j = R_j + (E_{i,j+1} - E_{i,j})$;	
5	end for	
6	while $ X > d$ do	$O(m-d)$
7	/* Start removing Voltage levels and measure the energy increase*/	
8	$k =$ index of smallest element in R	$O(m)$
9	remove V_k from X	
10	for all n cores such that $v(n) = V_k \Rightarrow v(n) = V_{k+1}$	$O(N)$
11	recalculate R_{k-1} & R_{k+1} ;	$O(N)$
12	end while	

Figure 1. RCM Algorithm

	V_1	V_2	V_3	V_4
C_1	8.1	8.3	8.5	10.5
C_2	4.3	4.7	5.6	8
C_3	6.3	7	7.5	10.3
C_4	∞	3.2	3.9	5.3
C_5	3.5	3.8	4.1	5.5
C_6	∞	2.8	3.2	4.6
C_7	∞	∞	2.4	3
C_8	∞	∞	∞	6.5

(a)

	1 st iteration	2 nd iteration	3 rd iteration
R_j	R_j	R_j	R_j
V_1	1.6	1.6	-
V_2	1.1	3.9	3.9
V_3	0.6	-	-
V_4	∞	∞	∞

(b)

Figure 2. RCM Example

D. Computational Complexity

As shown in Figure 1, the initial values for R_j require $O(m+N)$ runtime to compute, lines 1-5. Updating R_{j+1} and R_{j-1} , line 11, each iteration requires $O(N)$ to finish, since if voltage V_j is removed, only R_{j-1} and R_{j+1} are affected. The overall complexity of the algorithm is $O(m+N+(m-d)m+2(m-d)N)$. The best previous heuristic to solve the VSP has at least $O(m^2d^2N)$ complexity with near optimal solutions [7][8]. Both algorithms are linear with respect to N , however, we will show that practically, our algorithm is much faster, and computes solutions which are also near optimal for the considered cases.

IV. ENERGY OPTIMIZATION FRAMEWORK

In this section, we describe our energy optimization framework. This framework will be used to evaluate our voltage selection algorithm. RCM can be used in any problem that involves the Voltage Selection Problem. However, in this paper we use RCM in static mapping application on an embedded multicore considering PVT.

As shown in Figure 3(a), we assume an architecture

consisting of a grid of cores, each connected to a switch. Each switch is connected to four adjacent switches using two-way links.

A. PVT Impact

We use the PVT models described in [5][10]. Within-die process variation is modeled as multivariate normal distribution as shown in Figure 3(b).

We used HotSpot [13] to calculate the temperature during task scheduling. Since the accurate temperature value stabilizes after certain number of cycles and not at the start, the applications run at first using the ideal V_t and temperature values. Then the correct temperature T from HotSpot and the calculated V_t is plugged into the equations for delay and static power. Finally, a technology dependent resistive network is used to estimate the supply voltage variation [5]. The voltage variation due to the workload is calculated at each clock cycle during the application execution.

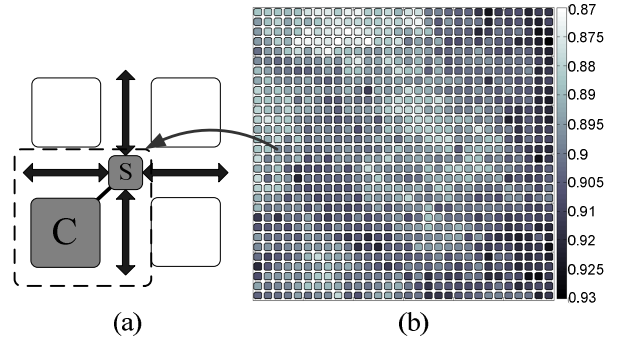


Figure 3. Architecture (a) core, its switch and links, (b) core's normalized frequency due to process variation

B. Energy Optimization Algorithm

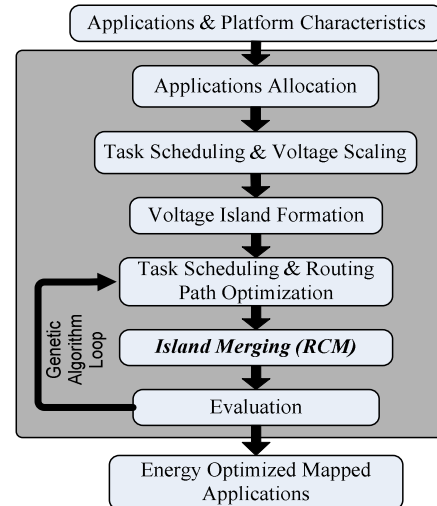


Figure 4. Optimization Flow graph.

Figure 4 shows the overall mapping algorithm. First, different applications are allocated portion of the

available resources. Tasks are then scheduled and assigned to cores using an As-Soon-As-Possible (ASAP) scheduling algorithm. The route at this point selected at random.

The slack of each task is computed, and used to scale the voltage of each core as low as possible such that all tasks can meet their deadlines. Voltage islands are then formed by grouping cores with the same voltage.

The routing traffic is then optimized using a genetic algorithm same as [5] and [6]. There are several possible routes for each data transfer. During the initial routing, each route is represented by a chromosome, and genetic algorithm (GA) loop optimizes each route. Further details can be found in [5] and [6].

Voltage selection using the RCM algorithm is then performed, and islands that are deemed to have the same voltage are then merged.

V. EXPERIMENTAL RESULTS

To validate our proposed RCM, we compared it to dynamic programming from [8], as it is the best method so far to give near-optimal results, and one step look ahead, 1-STEPLA, from [6]. We evaluated 50 test cases for each multicore. In every test case, a randomly generated task graph, with 450 tasks and value for parallelism, critical path length and number of edges within the range specified in Table I, [11], is mapped to a multicore with N cores [5].

Table I: Characteristics of the 50 test cases [11].

N# of Tasks	450 tasks
Parallelism range	6 to 48
Critical Path Length range	24 to 302 tasks
N# of Edges range	46 to 4749 edges

The number of cores considered in the simulation varies from 100 to 400 cores. Four voltage levels are selected from the range of 0.6V to 1.4V in 0.05V increments. Any unused cores are switched off completely. The technology considered in the experiment is 65nm with nominal $V_t=400\text{mV}$. The maximum speed at which the processor can run is 50MHz with the network running at 100MHz. The dynamic power is estimated to be 0.4mW/MHz. The energy optimization was carried out on a representative process variation (PV) profile [5]. The energy savings results of all 50 test cases are grouped and averaged.

Figure 5(a) shows the energy saving results for the three algorithms. Figure 5(b), shows the normalized runtime only for the competing methods, RCM and DP, written in MATLAB[®] and executed on Linux Debian with 2.8GHz ZION Intel[®] processor and 3G of RAM. Note that every data column in the figure is average of 50 cases. Although, RCM energy saving is identical to DP, RCM is more than 10x faster than DP. Given the big

advantage of RCM in terms of speed without losing any energy saving, RCM seems to be the best algorithm so far to deal with the VSP problem.

VI. CONCLUSIONS

In this paper, we proposed a novel approach to solve the voltage selection problem. We considered multi-core platform under the influence of within-die process, temperature, and voltage variations to show the superiority of our algorithm in speed as well as energy saving. We presented an energy optimization flow that uses voltage island technique to demonstrate our work. The voltage selection algorithm, Removal-Cost Method (RCM), provides much better energy savings than 1-STEPLA, slightly better savings than DP, with more than 10x speedup compared to DP.

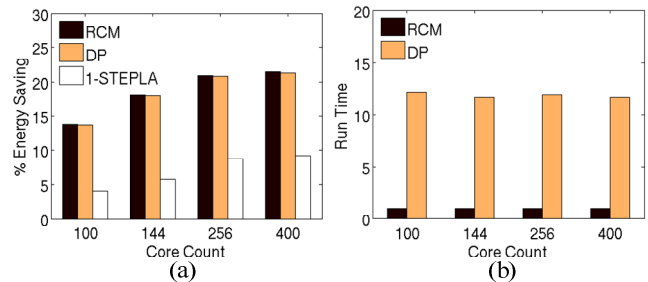


Figure 5. Experimental Results, (a) Energy Saving, and (b) Runtime

REFERENCES

- [1] A. Coskun, T. Simunic Rosing, K. Mihic, G. De Micheli, Y. Leblebici, "Analysis and Optimization of MPSoC Reliability," *Journal of Low-Power Electronics*, April 2006.
- [2] A. Andrei, P. Eles, Z. Peng, M. Schmitz, B. Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection," *IEEE TVLSI*, March 2007.
- [3] L. Yan, J. Luo, N. Jha, "Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-time Embedded Systems" *DAC*, 2003.
- [4] M. Ruggiero, A. Acquaviva, D. Bertozzi, L. Benini "Application-Specific Power-Aware Workload Allocation for Voltage Scalable MPSoC Platforms" *DAC*, 2005.
- [5] S. Majzoub, R. Saleh, and R. Ward, "PVT Variation Impact on Voltage Island Formation in MPSoC Design," *ISQED* 2009.
- [6] L. Leung, C. Tsui, "Energy-Aware Synthesis of Networks-on-Chip Implemented with Voltage Islands." *DAC* 2007.
- [7] P. Ghosh, and A. Sen "Energy Minimization using a Greedy Randomized Heuristic for the Voltage Assignment Problem in NoC" *SoCC*, 2008.
- [8] D. Sengupta and R. Saleh, "Application-driven Floorplan-aware Voltage Island Design", *DAC*, 2008.
- [9] E. Humenay, D. Tarjan, K. Skadron, "Impact of process variations on multicore performance symmetry." *DATE* 2007.
- [10] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas. "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," *IEEE TSM*, 2008.
- [11] T. Tobita, H. Kasahara, "A standard task graph set for fair evaluation of multiprocessor scheduling algorithms." *ICS'99*.
- [12] A. Das, S. Ozdemir, G. Memik and A. Choudhary. "Evaluating Voltage Islands in CMPs under Process Variations." *ICCD*, 2007.
- [13] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, M. R. Stan, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design." *IEEE TVLSI* 2006.